Giovanni M Guidini 16/0122660

Controle de Mesas de um Restaurante
Parte I

Relatório sobre o código-fonte do trabalho, detalhando as partes mais relevantes do código. Primeiramente será explanada a lógica geral do programa, seguida de um detalhamentos de todas as funções utilizadas, terminando com o singelo Easter egg adicionado.

Indice

1. Lógica Geral

O programa seguiu basicamente o fluxograma fornecido na especificação do trabalho. Foi implementado um grande loop para percorrer a entrada de dados, o processamento e a saída de dados, mas cada uma dessas partes foi separada em uma ou mais funções, para facilitar visualiação e possíveis alterações no código, conforme necessário. Pode-se dizer então que o programa utiliza da lógica *divide and conquer*.

Macros de pré-processador foram bem empregados. A utilização do comando #ifdef permite que o programa rode de forma consistente tanto em sistemas operacionais Windows como em Linux. Isso é necessario para definir o tempo de animação e o comando de limpar a tela.

O código conta com 3 variáveis globais: as 2 sugeridas na descrição do trabalho (tempo_saida e restaurante) e uma terceira (tip, do tipo double), usada para contar as gorjetas - explicado em Easter egg.

No total o programa conta com 9 funções além da main. Elas são: getControl, inicio, checkTables, getPeople, getPreference, sleep, animacao, freeTables, e easterEgg.

2.1 getControl

A função getControl é de retorno tipo void, com recebimento de um ponteiro para char (portanto por referência). Ela faz o teste de consistência do input para a continuidade do controle. Não possui nenhuma váriável além do parâmetro. Começa lendo uma string do teclado - uma string, e não um char apenas porque chars são muito sensíveis, e utilizando eles havia a possibilidade de um loop infinito no programa quando o usuário apertava diretamente enter para prosseguir o programa. Uma string resolve esse problema. - então a mensagem que pede input deve ser printada antes da chamada da função. Se o input não for adequado a função permanece num loop até que seja adequado.

Além desta funcionalidade principal getControl também chama a função easterEgg se receber o comando “tip” (outro motivo para usar uma string, e não um char). Para isso ela testa - somente na primeira vez que ler o input - se o controle fornecido é tip. Em seguida ela se chama novamente getControl e termina a execução da primeira função.

Linhas 39 à 51.

2.2 inicio

A função inicio, de retorno do tipo void, como o nome sugere, inicia as variáveis essenciais para o programa. Ela recebe um ponteiro para char pois já

chama a função getControl ao final de sua execução, iniciando propriamente o programa para recebimento de input do usuário. A função começa limpando a tela, definindo tip como 0, e atribuindo um novo valor para a seed. Em seguida vem a leitura da matriz restaurante e a definição de todas as mesas como livres no vetor tempo_saida. Finaliza mostrando a mensagem de boas vindas e perguntando ao usuário se ele deseja iniciar o gerenciamento, antes de chamar a função getControl e finalizar a execução.

Linhas 54 à 86.

2.3 checkTables

Esta é uma função essensial para o programa. De retorno tipo void, checkTables recebe um ponteiro para inteiro e faz a verificação das mesas do restaurante, guardando o número da mesa livre mais à esquerda e fazendo a chamada da função freeTables, para liberar as mesas que precisarem ser liberadas. Funciona desse jeito:

O ponteiro para inteiro, counter, é o contador da posição das mesas. Na chamada dessa função ele é um ponteiro para um vetor de inteiros de 2 posições, assim, este vetor guarda a posição da mesa mais à esquerda livre do salão em counter[0] e da varanda também em counter[1]. Isso é feito definindo as duas posições de counter como -1 e rodando um loop pelo vetor tempo_saida, para verificar se o tempo de cada mesa é menor do que o tempo do clock no momento da execução da função checkTables. Detalhe relevante: o loop passa por tempo_saida ao contrario (acessando a posição tempo_saida[15-i] no loop em que o indice i vai crescendo de 0 até 15), para que sempre fique armazenado em counter o menor número possível para a mesa.

Duas coisas podem acontecer durante essa verificação: primeiro, nenhuma mesa estar disponível, assim, o valor das duas posições de counter continuaria sendo -1, indicando que o restaurante está atualmente lotado, ou alguma mesa do restaurante está disponível e seu valor fica então armazenado na devida posição de counter.

Se a mesa estiver livre a função verifica se o valor de i é maior ou igual a 4 - se for, a mesa em questão pertence ao salão. Caso contrário, pertence à varanda. Verifica ainda se o valor do tempo naquela posição é diferente de 0. Se o valor for diferente de 0 significa que a mesa foi liberada agora (lembre-se que o tempo de todas as mesas é inicialmente declarado como 0 na função inicio), e a função checkTables então avisa que a mesa foi liberada e que a gorjeta dela está disponível. Em seguida chama a função freeTables para atualizar a imagem do restaurante. Outro detalhe inportante: para evitar que a função fique avisando que uma mesma mesa foi liberada várias vezes sem ser esse o caso o tempo da mesa recèm-liberada é definido como 0 no inìcio da função freeTables.

2.4 getPeople

Esta função, com retorno de tipo inteiro e sem recebimento de qualquer parâmetro pede a quantidade de pessoas que estão entrando no restaurante e faz o teste de consistência desse número. Como esperado, o número de pessoas apropriado é retornado. O teste é feito em um loop while simples. São aceitos grupos de 1 até 4 pessoas somente.

Linhas 120 à 133.

2.5 getPreference

Função com retorno de tipo inteiro e sem recebimento de parâmetros pede a quantidade de pessoas que irão para o restaurante e faz o teste de consistência do input fornecido.

Linhas 136 à 148.

2.6 sleep

Esta função foi fornecida na descrição do trabalho e simplesmente colada no programa. Trava a execução do programa por alguns milissegundos para criar a impressão de animação.

2.7 animacao

Muito do que o programa faz acontece nesta função. Ela é do tipo void, e recebe 3 parâmetros: ponteiro para inteiro (o array vago), e duas variáveis inteiras, n e pessoas.

A animação é dividida em 4 partes: 1. Da porta até o meio do salão, 2. Do meio do salão até a entrada do corredor apropriado, 3. Do corredor para a mesa certa e 4. A ocupação da mesa com o número certo de pessoas.

O esquema de animação é o mesmo para todas as partes: substituição de um caractere específico da matriz, limpa a tela, mostra a matriz, repete o número necessário de vezes.

A primeira parte é sempre igual, a partir da segunda que a função deve decidir o que fazer. Isso ocorre da segunte maneira: a variável n tem valor 0 (indicando preferência do salão) ou 1 (preferência da varanda). Quando a função animacao é chamada já é certo que a mesa de destino está livre. A posição n do array vago armazena o número da mesa de destino. Sabendo

qual a mesa de destino e em qual parte do restaurante ela fica é fácil mandar a party para o corredor certo. A linha do corredor é então armazenada em uma variável inteira k.

Depois vem a mesa. para isso utiliza-se a fórmula mesa = ((2* (vago[n]%4)) + (vago[n]%4)). Porque no resto da divisão por 4 o número da mesa pode ser 0, 1, 2 ou 3. Assim sabe-se até onde no corredor a party deve andar. obs.: a fórmula não é simplesmente o resto da divisão por 4 por causa do tamanho do corredor. Então no loop para a mesa a party está programada para ir até o final do corredor, mas parar caso a mesa já tenha chegado antes. A coluna da mesa é então armazenada numa variável l.

De posse das variáveis k e l - que dão a posição exata da mesa a ser ocupada na matriz - e do número de pessoas que está armazenada em (you guessed) pessoas, basta mudar os caracteres da mesa para ocupá-la.

A execução da função termina aí, mostrando a matriz atualizada com a mesa ocupada.

Linhas 156 à 291.

2.8 freeTables

Função do tipo void, recebe um único parâmetro inteiro mesa, o propósito desta função é atualizar a matriz mostrando as mesas recém liberadas sem as pessoas. No início da execução desta função o tempo_saida na posição mesa é atualizado para 0, evitando que freeTables seja chamada indevidamente.

Tendo o número da mesa - armazenado em mesa - a função descobre a linha k e a coluna l em que a mesa se encontra e retorna a mesa para sua configuração padrâo, independente de quantas pessoas estejam nela.

Ela também incrementa o valor de tip por valores pré-determinados dependendo da coluna que a mesa que foi liberada se encontra.

Linhas 293 à 337.

2.9 main

A função main faz aquilo que deveria fazer: controla o programa. Ela possui as variáveis inteiras pessoa, preferência e o array vago, e uma string controle. Antes de entrar no grande loop chama a função inicio. Estando dentro do loop ela faz as tomadas de decisão que irão dar prosseguimento as programa na seguinte ordem: Verifica se o controle fornecido é para dar continuidade ao gerenciamente. em caso positivo verifica se o restaurante está lotado (checando se os valores armazenados em vago[0] e vago[1] é -1). Se estiver avisa ao usuário e atualiza o restaurante. Se não estiver, pergunta o número de pessoas e a preferência de mesa. Em seguida acontece a verificação da disponibilidade da preferência de mesa. Estando disponível

ocorre a chamada da função animacao. Se não estiver disponível o programa oferece a outra opção e age de acordo com a resposta fornecida. No final pergunta-se se o usuário deseja continuar o gerenciamento. Se o usuário não deseja mais gerenciar o restaurante o programa mostra uma mensagem de agradecimento e exibe os ganhos totais da noite, e encerra o programa.

3.0 Das gorjetas

A ideia das gorjetas era deixar o programa mais divertido, basicamente. O valor das gorjetas começa em 0 e vai sendo aumentado para cada mesa liberada. É uma variavel do tipo double, com nome tip, e é global porque existem 4 funções que precisam conhecer essa variável. Além disso, é uma variável completamente irrelevante para a funcionalidade do programa, então não havia necessidade de ficar enviando-a como parâmetro para as funções que precisam dela. Isso iria distrair quem precisar alterar o código do real propósito do programa. Assim, ela pode ser vista por qualquer função e é utilizada quando conveniente.

A variável tip é modificada na função inicio (quando recebe o valor inicial 0) e na função checkTables (onde é incrementada). Na função getControl o usuário tem a opção de fornecer o comando tip. Isso irá mostrar os ganhos da noite até aquele momento. Quem faz isso é a função easterEgg, que tem esse nome somente porque não é dito ao usuário que o comando tip existe. No encerramento do programa o valor de tip é exibido como os ganhos totais da noite.